Prepared by: Shubham Patel
Role: Azure Cloud Engineer
Date: 26 June 2025

Cost Optimization Assessment: Managing Billing Records in Azure Serverless Architecture

Executive Summary:

This assessment presents a strategic solution to reduce escalating costs in an Azure serverless architecture where billing records are stored in Azure Cosmos DB. While the system is read-heavy, records older than three months are rarely accessed—yet continue to consume costly storage and throughput.

To optimize expenses without disrupting service, we propose a seamless archival strategy that offloads infrequently accessed data to Azure Blob Storage. This hybrid approach ensures cost efficiency, zero downtime, unchanged APIs, and full data availability with minimal latency.

Problem Analysis:

The system stores over 2 million billing records in Azure Cosmos DB, each up to 300 KB. While the architecture is read-heavy, records older than three months are rarely accessed. Despite this, all data remains in high-cost storage.

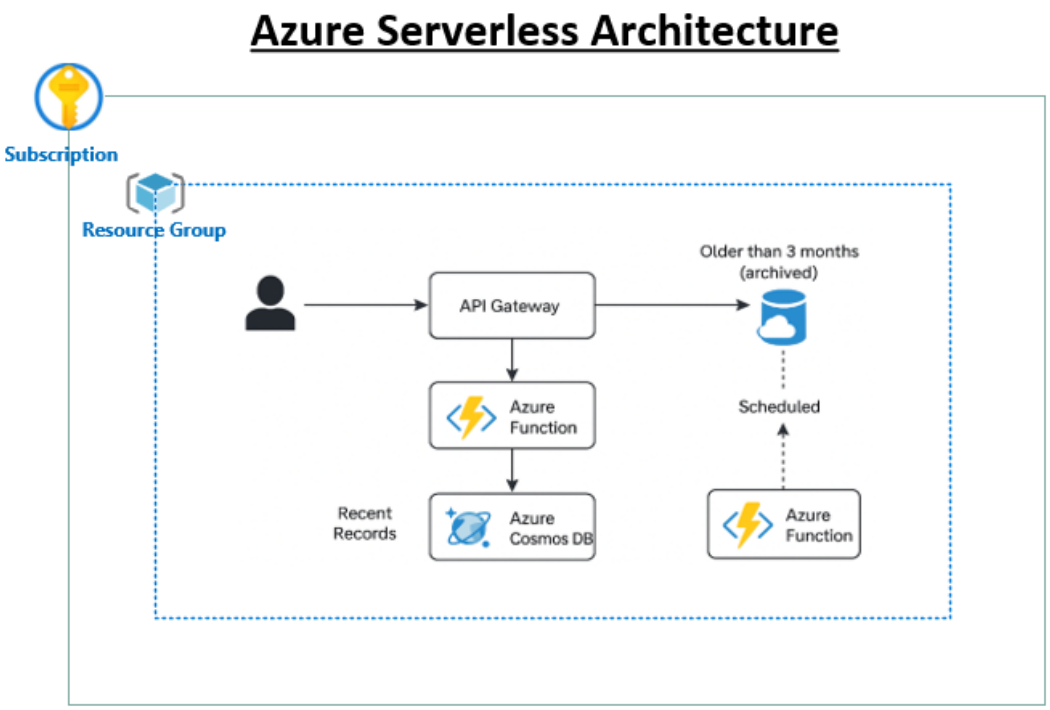
This results in rising costs due to unnecessary storage and throughput (RUs) for infrequently used data. However, we must ensure:

- Fast access to old records (within a few seconds),
- Zero downtime or data loss,
- No changes to existing APIs.

Proposed Solution Overview:

We propose a hybrid approach:

- Keep recent records (last 3 months) in **Cosmos DB** for fast access.
- Archive older records to **Azure Blob Storage** to reduce costs.
- Use an **Azure Function** as a smart proxy to fetch data from either source, ensuring APIs remain unchanged.



Implementation Strategy:

The solution is implemented in three key phases:

1. Archiving:
A scheduled Azure Function identifies records older than 90 days, converts them to JSON, and stores them in Azure Blob Storage. The original entry is then deleted from Cosmos DB.

2. Retrieval:
The read API is routed through a Function Proxy. It first queries Cosmos DB; if the record isn't found, it seamlessly fetches from Blob Storage—ensuring no API changes.

3. Storage Format:
Archived records are organized as:
`billing-archive/YYYY/MM/record-id.json`
This structure ensures fast, efficient retrieval.

Jason:


```
{
 "id": "BR-948123",
 "customerId": "CUST-45678",
 "amount": 1299.50,
 "currency": "INR",
```

```
  "billingPeriod": {
    "start": "2025-05-01",
    "end": "2025-05-31"
  },
  "generatedOn": "2025-06-01",
  "archived": true
}
```

Pseudocode:

```
def archive_old_records():
    cutoff_date = today() - timedelta(days=90)
    old_records = cosmos_query("SELECT * FROM Billing WHERE timestamp < @cutoff", cutoff_date)
    for record in old_records:
        blob_path = f"billing-archive/{record['timestamp'].year}/{record['timestamp'].month}/{record['id']}.json"
        blob_client.upload(blob_path, json.dumps(record))
        cosmos_delete(record['id'])
```

Reading with Smart Fallback

```
def get_billing_record(record_id):
    try:
        return cosmos_get(record_id)
    except NotFound:
        blob_path = find_blob_path(record_id)  # e.g., search index or use metadata
        return blob_get(blob_path)
```

Benefits Summary (Short)

- **Lower Costs**: Moves old data to cheaper storage.
- **No Downtime**: Runs without interrupting services.
- **No API Changes**: Seamless integration.
- **Fast Access**: Archived data remains quickly accessible.
- **Scalable & Simple**: Easy to implement and maintain.

Taken Help with chatGPT

- How does this design scale with 100 million+ records, and what adjustments are needed to sustain performance and cost-efficiency?

- What are the implications of eventual consistency in Cosmos DB during archival, and how can we ensure data integrity during transitions?

- How do we ensure the archival process is idempotent and fault-tolerant, especially during retries or partial failures?

- How do we maintain read performance from Blob Storage as the volume of archived records grows significantly?

- Can this architecture be made multi-tenant without major redesign, and what changes would be required to support tenant isolation and security?